**Problem 1.**

a. Assume T = (V, S) and T' = (V, S') are two different MST for G

e is the unique cheapest edge that is one of only T or T’.

Without loss of generality, assume e is in T.

We have: w(e) >= w(f) every edge f on the path from x to y in T' (Since if w(e) < w(f), then w(T’f->e) < w(T’), contrary to T’ being a MST)

But because edge weights are distinct, w(e) > w(f). By the way e was chosen, every edge on the path in T' from x to y is in T as well. But these edges of T, combine with the edge e of T, form a cycle, contrary to T being a tree.

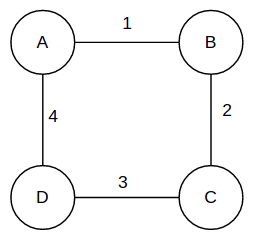
So G has an unique MST.

b. Suppose for contradiction that there is a MST T that does not contain the lightest edge e from u->v. Since T is a connected tree, therefore adding the edge e (u, v) to T forms a cycle. Following this cycle, there must be at least one other edge in the cycle crossing the edge e. Removing this edge, and leaving (u, v), we have a tree with a lower total weight than the original T. But T was assumed to be an MST, this is a contradiction.

So, lightest edge e must be part of every MSTs.

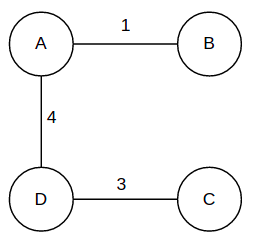
**Problem 2.**

a. No, For example, Graph G connected, undirected:



The bottleneck edge is C-D, all the spanning trees that contain the edge C-D are the minimum bottleneck spanning trees.

This tree below is a minimum bottleneck spanning tree but not a minimum spanning tree



b. Suppose T be the minimum spanning tree of a graph G(V, E) and T’ be its minimum bottleneck spanning tree. Consider the maximum weight edge of T and T’(bottleneck edge).

Then, there are three cases possible:

Case 1: If both the edges is the same. Obviously, T is MBST as well. Every MST is MBST

Case 2: If they are different and the maximum weight of T’ is greater than the maximum weight of T. That cannot happen to MBST, so T has to be a MBST.

Case 3: If they are different and the maximum weight of T is greater than the maximum weight of T’. Let assume maximum weight edge e from u->v is in T (not in T’), then weight of e > weight of every edges in MBST.

Let X is a cluster of all vertices in T that can be reached from u without through v.

Let Y is a cluster of all vertices in T that can be reached from v without through u.

There have to be a cut edge connected between X&Y. It must be the minimum weight.

But (u, v) is the minimum edge since T is MST. So this is a contradiction.

So every MST is MBST.

**Problem 3.**

Step 1: Sort the edges of G follow decreasing order by weight. Let T be the set of edges with the maximum weight in spanning tree.

Initially T = ∅.

Step 2: Add the first maximum weight edge to T.

Step 3: Add the next edge(in sorted array) to T so that does not form a cycle in T. If there are no remaining edges exit and report G to be disconnected.

Step 4: Loop step 3 until T has n-1 edges (n is the vertex number in G) stop and output T.

**Problem 4.**

i.

Step 1. To find the optimal coding, We need to know frequency of each character in the string, that means calculate frequency of each character.

Step 2: The purpose to build a binary tree, each unique character as a leaf node. Sort the characters in increasing order of the frequency.

Step 3: Create new node that its value is sum of 2 minimum frequencies.

Assign the minimum frequency to the left child a the second frequency to the right child

Step 4: Then treat that node as a new character with frequency value and remove the 2 minimum value from sorted array in step 2.

Step 5: Repeat step 2-4 until remains only 1 value in sorted array.

The tree that is built in step 3,4 is the thing we need, treat 0 as left edge and 1 as right edge.

The characters (leaf nodes) is assigned to a binary from root node to the leaf.

ii.

Huffman code is the optimal solution, we can prove that:

In Huffman algorithm, there is no way for a more frequent symbol smaller codeword. The codeword length is ordered follow as the the frequency of that character.

So Huffman coding is optimal.

iii.

Step 1. Start from the root node.

Step 2. If the current bit in the given data is 0,then move to the left node of the tree.

Step 3. If the current bit in the given data is 1,then move to the right node of the tree.

Step 4. During the traversal if leaf node is encountered then print character of that leaf node.

Step 5. Loop of the encoded data starting from step 1 until all characters is decoded.

**Problem 5.**

There are 4 cases when update cost for an edge, we can design an algorithm like this:

switch (Edge e):

case 1: Edge e is in MST and cost w decreases → MST is not changed

case 2: Edge e is not in MST and cost w decreases,

Add e to current MST, now MST+e → 1 cycle

Remove the highest value in that cycle → new MST

case 3: Edge e (vertex u->v) is in MST and cost w increases

Remove that edge from MST → 2 components(not included edge e) that connected

(Now, we need to find an edge with smallest weight to connect these 2 components)

iterate all edges to find which edge connect 2 components with smallest weight.

Add that edge to form new MST.

case 4: Edge e is not in MST and cost w increases → MST is not changed